Software Automated Testing Guidelines

Malik Jahan Khan, Abdul Qadeer, Shafay Shamail
Department of Computer Science
Lahore University of Management Sciences (LUMS)
Lahore – Pakistan
jahan@lums.edu.pk, qadeer@lums.edu.pk, sshamail@lums.edu.pk

Abstract
Today software marketplace is competitive and volatile like never before. Commercial software complexity is increasing while time pressure to release high quality software is also on the rise. On one hand more defects are probable due to increased software complexity while on the other hand organizations cannot afford to let slip many bugs in the field. In an effort to test an application reliably, quickly and thoroughly, automated software testing has emerged as a viable solution to meet today's challenges. Although it is true that automated software testing tools have become very powerful, it should be kept in mind that proper induction and utilization of tool is also nothing less than a challenge. The primary focus of this paper is to describe the common pitfalls and to recommend guidelines to maximize the success probability of tool induction.

1. Introduction
Most of today's enterprise software is developed using some variant of agile process like RUP [8]. The main idea of these processes is to break-up the whole big project into many small manageable parts. Each part is released to the client, while the subsequent releases integrate with the older one. These kinds of processes have their own benefits for all the stakeholders.

Once a project is complete and the whole product is fully deployed, the product goes into the maintenance phase. Software in general and enterprise applications in specific should not resist changing. Due to this reason these days software are built using pluggable components. Any component could be changed anytime as the needs arise. There could be numerous reasons for the change. We are not discussing the reasons to initiate these changes but one thing is obvious that in today's volatile world change is inevitable in the software.

In the above two scenarios, it is notable that software testing team has to do lot of regression testing. In scenario one, whenever a new iteration is complete and it is merged with the previous release, the inspection team has to thoroughly test the new functionality while running the regression test on the previous release, to make sure that integration is smooth. In a big enterprise application the amount of software, to be regression tested, increases. In scenario two, mostly the client changes are too small as compared to the size of the whole project. Again the result is that once the change is implemented, inspection team has a lot of testing work as far as regression testing is concerned. Although software organizations try to circumvent this situation by intelligently analyzing and segregating software parts which could potentially malfunction and hence are good candidate of regression testing. But this method has its potential risks.

Iterative development and changes initiated by client are not the only situations when testing team has lot of work to do. Different organizations perform different testing cycles on their releases. Similarly after each bug fix, localized regression is performed. So the workload of inspection team is ever increasing [5].

In this paper, we are not discussing that how an organization could choose from its repertoire of choices. Our focus is to present guidelines for an organization, which has already decided to go for automated testing. Automated software testing is comparatively a newer approach of testing and lot of myths surrounds this technique. In section 2, we have discussed various testing alternatives. Automated testing is one of them. In section 3, we have discussed some benefits of automated testing. In section 4, a few challenges of automation have been discussed briefly. In rest of this paper, a prescription or guidelines for an organization which has already arrived at the decision to use automated testing, have been suggested in detail.

2. Testing Alternatives
To meet a certain quality level, an organization could have the following alternatives at its disposal:

2.1 Commit their inspection team to work longer and harder
Our local software industry is notorious for longer working hours. Ask any software engineer and he

will tell you his wish to keep a balance between his personal life and work. It is very common for organizations to commit their current human resource to work longer and harder. But this could only serve as a short term solution. There is always the hazard that people will burn out and the turnover rate could increase due to people leaving the organization. In any case there is always high probability of missed bugs, slipped into the field.

2.2 Increase the personnel in the inspection team

The second option is to increase the testing personnel in the inspection department. Organizations need to conduct cost benefit analysis before hiring new resources. Hiring experienced resources is more costly while hiring fresh graduates and training them in-house has its own cost and risks. The direct costs are in terms of pays and trainings. The in-direct cost is that the experienced people have to devote their time to train new resources. There is always some risk associated, if new resources are put on work without giving them enough time to understand the business domain of the software.

2.3 Give more testing time to the same team

The third option is to give the same testing team more time, so that they could test properly. But it would be against the spirit of an agile process if system testing is consuming a big chunk of the total iteration time. Software business is very competitive these days and organizations are looking ways to expedite the software delivery. So this option is no more an option for most of the organizations.

2.4 Do not do anything and rely on whatever testing is possible in given time

Option four is again not viable. These days, organizations have to be proactive to stay profitable in the business. It should be noted that using this option will result in decreased product quality by slipping the bugs in the shipped release. Later, these bugs would be reported by the client. These field defects not only reduce customer satisfaction but also put the organization under pressure as they now have to take care of bigger backlog of field bugs along with new release. So neglecting problems is no solution because problems will keep on coming back with more intensity.

2.5 Use automated testing

The fifth option is to use the automated testing. Automated testing is a technique to test software by means of a software tool, which records the user behavior and later it could replay the recorded script to mimic or simulate the human behavior. The recorded script could be enhanced by means of programming. There are different automation tools for white box and black box testing. For example Rational's Purifier is used to detect any memory leaks in C / C++ code. Mercury Interactive's WinRunner and Quick Test Professional (QTP) are examples of GUI based functional black box testing.

3. Benefits of Automated Testing

Automated testing is a very powerful technique. But to extract this power proper induction and usage of tool is mandatory. Let us discuss the major benefits of an automated testing tool.

3.1 Fast

Although automated testing tools mimic or simulate a human user but they do it very quickly as compared to a human being. In manual testing the rate of test case execution varies due to many human factors like a person is not equally productive the whole day. On the other hand an automated tool could run the script with the same high speed each time, it is executed. It dramatically decreases the testing time and hence utilizing the inspection time to the fullest.

3.2 Reliable

In manual testing, there is always a chance of human error. Automated testing tools, on the other hand, run the scripts reliably each time. Exact same steps are followed every time, the script is run.

3.3 Repeatable

The recorded script is repeatable. One can test how a website or application reacts after repeated execution of the same operations. Repeated execution becomes very cumbersome in manual testing.

3.4 Programmable

In almost every script, the recorded script is enhanced by means of programming elements. These programming elements enhance power of the scripts. Now they can verify a larger range and variety of functional requirements.

3.5 Comprehensive

One can build a suite of tests that covers every feature in a website or application, hence, increasing the code and data coverage. It is always desirable to test the complete functionality of the software.

3.6 Reusable

One can reuse tests on different versions of a website or application, even if the user-interface changes.

4. Challenges of Automation

Though automated testing has a lot of benefit, but it also has some associated challenges. Following list includes some of the major challenges of automation. Our suggested guidelines will be helpful to face these challenges [3]:

i. Selection of Test Tool
ii. Customization of Tool
iii. Selection of Automation Level
iv. Development and Verification of Script
v. Implementation of Test Management System

5. Guidelines [2], [7], [1]

Many expectations are not completely met in automated testing, but it really adds a lot of benefits, even done by right people in right environment at right time [6]. Cost/benefit analysis is very important before going for automated testing. Adapting automated testing for trivial and small cases may be quite expensive. So, experienced decision makers play an important role here. Following suggested guidelines will be quite helpful to adapt a successful automation.

5.1 Technical and Managerial Decision

Incorporating test automation could be a technological decision at the beginning but ultimately technical managers have to defend their proposal in front of organizational management. Different people have their own point of views about different things. The gap between different people could be wider depending on different circumstances. So it is very important to discuss the automation decision inside out to reach a common understanding among stake holders. Time, cost and resource requirements should be crystal clear among the management levels.

5.2 Planning Automated Testing

Appropriate plan is required to introduce automated testing in the organization. This plan should have measurable milestones, shared with the upper management as well.

5.3 Market Survey

Market survey is necessary. This market survey has two dimensions. Firstly the organization should thoroughly know the kind of software they want to apply automation on. They should also get automation success and experience of other organizations, working in similar domain. Mostly other organizations don't provide such data. So company has to invest either for a related consultant or they need to buy the industry survey reports. It should be noted that there are many automated software testing vendors in the market and it may not be feasible for the organization to try every product to decide among the tool. It is highly recommended that an organization try to buy a comparison report of automated tools. In market, every vendor claims that its tool is easy to use, friendly, and reliable and your non-technical people can easily learn and use it [9]. But you need to evaluate after a comprehensive survey and then select the really stable tool.

5.4 Proof of Concept

Once the tool choice is narrowed down (not more than two), the next task is proof of concept. It is very important to make a dedicated team to work on proof of concept. Their task is to verify the tool capabilities by having first hands on experience on the organization software. Few automated software are available for limited time trial version. Some times, organizations need to buy at least one license for this purpose. This phase is very crucial because company decision greatly depends on the results of this study. So it goes without saying that this group has the responsibility to appropriately selecting the portions of software to be tested using the tool, which are representative of whole application.

5.5 Trained People

To conduct proof of concept, company need some people who know automated testing. So company could either hire people or could train their resources for automated testing. To train people could be a costly option, but to conduct proof of concept, company needs people who are with the organization for quite some time and they understand the company software domain well.

5.6 Automated Test Script

Writing automated test script is just like any other development activity. The difference is that the requirement gathering and high level design is not needed. Design, coding and testing of automated test script code is very important. Don't underestimate this point because consider how sarcastic it would be that an organization inducted an automation tool to increase their testing efficiency but they ended up with more bugs slipped into the field because the automated script wasn't able to detect it due to a bug in its script. This is a major cost of test automation. Again the test script writer isn't the best person to test it.

5.7 Properties of Test Script

Test script needs all the qualities of a good software like maintainability, proper documentation etc. This is because if test scripts could not be changed quickly and easily, then they are almost useless. All this needs experienced test script writers. It is recommended that the organizations identify the portions of the program, which are least probable to change and yet important for regression test. If they

couldn't find such functionality and they don't have experienced test script writers, it is recommended not to do test automation at this point.

5.8 Significance of Test Case

It is a myth that automated testing can catch more bugs as compared to manual testing. It is necessary to understand that an automated test script could only be as efficient in finding bugs as the original test case. Test script is a translation of a test case. If a test case skips something, the automated test script will miss it.

5.9 Manual Test Cases

For test automation, detailed test script along with test data is necessary. Some times, a level of abstraction hides the detail of test case because the testing engineer has the domain knowledge. If automated test script writer is someone else, the proper detail is necessary. If the organization could train all of their test team for automation, so that they could write the test script for their own test cases, that is the ideal case. If it is not possible and there are only few people who know automation well and they have to automate everyone's test cases then it is recommended to increase the quality and detail of manual test cases. The automated test script should be tested by the person, who originally wrote the test case.

5.10 Significance of Automation Tool

Give your automation tool and yourself sufficient time before judging its usefulness. Although it is very important to quantitatively measure the efficiency due to test tool but the conclusions should be made after sufficient time. The organization should rely on quick tutorial based tool learning approaches. Vendors provide very good tutorials with tools. The best bet is to put the people to write the real test script after learning most important and basic features. Advanced features could be learned as needed. It decreases training cost. The automated test scripts may not have all those nice features like maintainability in the beginning, but after some time, organization would have valuable work force. One important factor is to hire people in inspection team with right qualification, so that they have programming background. This thing really increases the speed of script coding.

5.11 Requirements Change

Widespread application change could make all the test scripts useless and they may need to change. If some part of software is highly probable to be changed, it shouldn't be automated.

5.12 Incorporating Intelligence

Automated testing tools can not mimic human intelligence. When a test engineer manually executes the steps of a test case, s/he could intelligently change his/her actions as needed. Incorporating such customization in the script is very difficult, but not impossible.

5.13 Exhaustive Testing

Exhaustive testing is yet not possible. Although code and data coverage is greatly increased, but yet testing each branch of code with each permutation of data is not possible.

5.14 Time to Run Automated Test Tool

In GUI-based automated test tools, it is difficult to start writing scripts if code is yet not ready. That means, it may not be possible to use the tool from the very beginning of a project. The best situation is when first release is manually tested and deployed. Then while release two is under development, the automated scripts of release one should be written for the purposes of regression testing.

5.15 Testability

Testability becomes one of the most important traits after the induction of a costly tool (The cost of a sophisticated tool is around $4000 per license). Testability is the quality of a software that how easy or difficult it is to test the software. Many third party components may not be recognized by the tool. In these conditions organization should provide the developers with a list of GUI gadgets, which could be recognized by the automation tool.

5.16 Automation Success

The Software Test Life Cycle (STLC) is the roadmap to automation success. STLC parallels the Software Development Life Cycle (SDLC), and includes phases planning, analysis, design, construction, testing (initial test cycles), bug fixes and re-testing (final testing and implementation) [4]. Lack of planning earlier is one of the biggest reasons of automation failures. A lot of automation work should be done before the software is available. "Start Early" is the big lesson, practitioners recommend for successful automation [4].

5.17 Part of Culture

Make the automation part of culture. Though, it can't be guaranteed to experience a successful automation in the first go, however, "experience and learn" leads to successful adaptation of this phenomenon. Finally, this success will make automation part of the organizational culture. The

complex and hardly controllable future projects need a pre-existing culture of automation.

The computer which runs automated scripts may not be used in parallel for other activities. It is a misconception that a test engineer will manually test application while the tool is running its script in the background. When the automated tool mimics the user behavior by clicking the gadgets, the operating system thinks that it's the active window and hence brings that window in the foreground. That means that a test engineer may not do something else, while test script is running in the background. So either separate machines are required or test engineer may need to observe running script. Some times, un-attended scripts are run at the close of business to run over night. But again writing unattended scripts is more difficult than ordinary scripts.

6. Conclusions

Every organization is unique and has its own needs. It is difficult to recommend "one size fit all" solution. But following the guidelines given in this paper increases the probability of success and could make the automation experience satisfactory for all the stakeholders. These guidelines might prove to be quite effective when organizations customize them according to their own business architecture, available human resources, nature of the project, level of quality needed and budget constraints.

Acknowledgements

We are highly thankful to Tauqeer Hussain, Tariq Fayyaz and Zeeshan Ali Rana for their valuable help and guidance towards completing this work.

References

[1] AutoTester Inc. *Best practices guide to automated testing*, A document of basic description of Automated Testing, http://www.autotester.com/content/pdfs/BPG-pdf.PDF, 2002.

[2] Dustin, E. *Lessons in test automation*, Software Testing & Quality Engineering, September/October 1999 pp. 16-21.

[3] Hughes Software Systems Ltd. *Test Automation*, http://www.hssworld.com/whitepapers/whitepaper_pdf/test_automation.pdf, December 2002.

[4] Isenberg, H. M. *The Practical Organization of Automated Software Testing*, http://www.automated-testing.com/PATfinal.htm, 2005.

[5] Kan, S. H. *Metrics and Models in Software Engineering*, Second Edition, Pearson Education, 2002.

[6] Kerry, *Automated Software Testing – A perspective*, http://www.testingstuff.com/autotest.pdf

[7] Pettichord, B. *Seven steps to test automation success*, http://www.io.com/~wazmo/papers/seven_steps.html, June 2001.

[8] Pressman, R. S. *Software Engineering, A Practitioner's Approach*, Sixth Edition, McGraw Hill Education, 2005.

[9] Zambelich, K. *Totally Data-Driven Automated Testing*, A White Paper of Automated Testing Specialists Inc., http://www.sqa-test.com/w_paper1.html, 1998.